VOL. 16 NO. 3 SEPTEMBER 2015

ISSN : 1411-3201

Jurnal Ilmiah DASI

DATA MANAJEMEN DAN TEKNOLOGI INFORMASI

STMIK AMIKOM
YOGYAKARTA

VOL. 16 NO. 3 SEPTEMBER 2015
JURNAL ILMIAH
Data Manajemen Dan Teknologi Informasi

Terbit empat kali setahun pada bulan Maret, Juni, September dan Desember berisi artikel hasil penelitian dan kajian analitis kritis di dalam bidang manajemen informatika dan teknologi informatika. ISSN 1411-3201, diterbitkan pertama kali pada tahun 2000.

PENANGGUNG JAWAB :
Ketua STMIK AMIKOM Yogyakarta, Prof. Dr. M. Suyanto, M.M.

ALAMAT PENYUNTING & TATA USAHA
STMIK AMIKOM Yogyakarta, Jl. Ring Road Utara Condong Catur Yogyakarta, Telp. (0274) 884201 Fax. (0274) 884208, Email : jurnal@amikom.ac.id

BERLANGGANAN
Langganan dapat dilakukan dengan pemesanan untuk minimal 4 edisi (1 tahun) pulau jawa Rp. 50.000 x 4 = Rp. 200.000,00 untuk luar jawa ditambah ongkos kirim.

DAFTAR ISI

UJI PERFORMA IMPLEMENTASI SOFTWARE-BASED OPENFLOW SWITCH BERBASIS OPENWRT PADA INFRASTRUKTUR SOFTWARE-DEFINED NETWORK

Rikie Kartadie[1)], Barka Satya[2)]

[1)]Teknik Informatika STMIK AMIKOM Yogyakarta
[2)]Manajemen Informatika STMIK AMIKOM Yogyakarta
email: r.kartadie@amikom.ac.id[1)], barka.satya@amikom.ac.id[2)]

Abstract

Software-based OpenFlow switch performance is still untested, so it still can be a reference / recommendation that the right to be able to replace the precious dedicated high OpenFlow switches. This study examined the performance of software-based switches OF by comparing it with a dedicated OpenFlow switches presented by mininet and with non OF switches. In general, the research consists of several steps, namely simulated using an emulator mininet as a comparison test (as representative of a dedicated switch OpenFlow), testing the performance of the switch OpenWRT which has not given OpenFlow agent (switches non-OF), performance testing of software-based switches OF. Switch OF-based software can be used to replace a dedicated OpenFlow switches to implement SDN well on medium-scale and campus based throughput and jitter test results performed.

***Keywords*:**
Switch Openflow Software-Based, Software-Defined Network, Uji Performa

Pendahuluan

Perkembangan pesat *Software-Defined Network* telah dirasakan oleh vendor-vendor besar. HP, Google dan IBM, mulai merubah pola *routing-switching* pada jaringan mereka dari pola *routing-switching* tradisional ke pola infrastruktur *routing-switching Software-defined Network(SDN)*. Pengiriman data yang besar dan kompleks pada infrastruktur *enterprise*, menimbulkan beberapa masalah, diantaranya apakah data tersebut dapat dikelompokkan berdasarkan aplikasi yang dijalankan. Pada kenyataannya, infrastruktur *enterprise* memiliki banyak jalur yang tentu saja akan melewatkan banyak data, mulai dari *email, e-commerce, web, social engineering, video, voice* dan masih banyak lagi, data-data tersebut akan berjalan bersamaan dan tidak ada pengaturan yang khusus untuk itu, pengaturan aliran data hanya sebatas pada sistem *routing, QoS* dan pengkotakkan seperti *packet filtering* dan ACL yang diterapkan pada perangkat masing-masing vendor. Dengan *Software-Define Network / OpenFlow*, kita dapat mengelompokkan data tersebut dalam kelompok-kelompok kecil secara virtual, sehingga terkesan bahwa data email hanya berjalan pada jaringan email, data web hanya berjalan pada jaringan web dan seterusnya.

OpenFlow adalah protokol yang relatif baru yang dirancang dan diimplementasikan di Stanford University pada tahun 2008. Protokol baru ini bertujuan untuk mengontrol data plane switch, yang telah dipisahkan secara fisik dari *control plane* menggunakan perangkat lunak pengendali (*Controller*) pada sebuah server. *Control Plane* berkomunikasi dengan *data plane* melalui protokol OpenFlow. Bentuk *Software-Defined Networking (SDN)* memungkinkan para peneliti, administrator dan operator untuk mengontrol jaringan mereka dengan perangkat lunak khusus dan menyediakan *Application Programming Interface (API)* terhadap tabel forwarding dari switch dari vendor yang berbeda .[1]

Selain itu, Shirazipour , M., dkk menyatakan bahwa saat ini OpenFlow merupakan satu-satunya standar yang tersedia dan diterima secara luas untuk protokol SDN [2]. Fungsi OpenFlow dan SDN telah dipelajari dalam beberapa tahun terakhir untuk *packet networks* (paket jaringan), tetapi implementasinya masih terbilang jarang, Sherwood,R, dalam presentasinya mengatakan hanya ada 8 Universitas dan 2 badan riset nasional (*National Reserch Backbone*) yang telah menggunakan OpenFlow. [3]

Untuk melakukan eksperimen tentang OpenFlow, para peneliti sering kali harus menggunakan perangkat *hardware/dedicated switch OpenFlow* yang dikeluarkan oleh beberapa vendor dengan harga yang tinggi. Selain itu ada pula jenis switch *OpenFlow software-based switch OpenFlow* dengan basis OpenWRT dengan harga yang lebih terjangkau. Kenyataannya, *software-based switch OpenFlow* (selanjutnya dalam penelitian ini diguna-kan istilah *Switch OF software-based*) masih belum teruji performanya, sehingga masih belum dapat menjadi acuan/rekomendasi yang tepat untuk dapat menggantikan perangkat *dedicated switch OpenFlow* yang berharga tinggi.

Berdasarkan latar belakang diatas, dapat dirumuskan beberapa permasalah yang akan diangkat dalam penelitian ini : (1) Bagaimana performa *switch OpenFlow(OF) software-based* bila dimplementasikan menjadi infrastruktur *Software Defined Network.*(2) Dapatkah *switch OF software-based* menggantikan perangkat *dedicated switch OF (hardware-base)* yang mahal, dalam untuk implementasi SDN.

Tinjauan Pustaka

Menurut Chunk Y. EE. OpenFlow dapat dimplementasikan pada NetFTPGA, dijadikan sebuah *firewall* dan dapat dimonitoring menggunakan *wireshark* [4]. Perbedaan mendasar pada penelitian ini adalah pada perangkat penelitian yang digunakan, penelitian ini menggunakan *switch OpenWrt* yang langsung terhubung dengan kontroler, sehingga bila ternyata berhasil, implementasi pada infrastruktur jaringan tidak menjadi beban biaya yang besar, karena tidak terjadi perubahan yang mendasar pada *hardware* infrastruktur yang telah ada.

Applement, M. dan De Boer, M. melakukan analisis performa terhadap *hardware openflow*. *Hardware openflow* adalah seperti *NetFPGA card, Pica8 OpenFlow on a Pronto switch* dan Open vSwitch. Pengujian dilakukan pada beberapa variabel diantaranya *QoS, Port Mirroring, fail over speed* dan *performance overhead* [1]. Perbedaan mendasar pada penelitian ini adalah dari jenis *hardware* yang digunakan, penelitian ini menggunakan *software-based openflow switch* dengan *platform* openwrt.

Dalam tesis Kartadie, R., melakukan penelitian dengan menggunakan 2 buah switch OF *software-based* dan dibandingkan dengan emulator mininet untuk melihat keberhasilan prototipe yang dibuat, namun belum diteliti tentang performa dari switch yang dikerjakan bila diimplementasikan ke infrastruktur berskala besar [5]. Penelitian ini meneliti performa dari *switch OF software-based* dengan mengimplementasikannya pada topologi yang digunakan pada infrastruktur SDN.

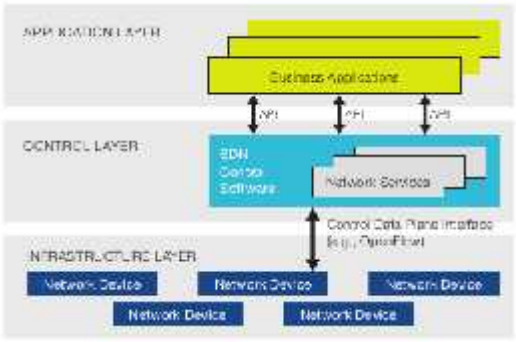
Landasan teori

Infrastruktur *Software-Defined Network*

Konsep *Sofware-Defined Network* (SDN), pertama sekali diperkenalkan oleh Martin Casado di universitas stanford pada tahun 2007 dengan tulisan pada jurnalnya berjudul "*Ethane: Taking Control of the Enterprise*". Pada jurnal tersebut dikatakan bahwa ethane adalah sebuah arsitektur baru untuk perusahaan, yang mengizinkan manajer mendefinisikan luasan dari sebuah jaringan, kebijakan jaringan dan kemudian menjalankannya secara langsung. Ethane adalah sebuah penyeder-hanaan yang ekstrim dari ethernet *switch* dengan sebuah kontroler terpusat yang mengatur hak masuk dan aliran routing. [6]

Software-Defined Networking (SDN) atau split arsitektur adalah sebuah konsep yang memungkinkan/memperbolehkan operator jari-ngan untuk mengelola *router* dan *switch* secara fleksibel menggunakan *software* yang berjalan di server eksternal [2], yang dikutipnya dari IETF *Working group*, "*Forwarding and control element separation (forces) framework*" http://www.ietf.org/html.charters/forcescharter.html .Sedangkan *Open Network Foundation*, mendefinisikan SDN adalah sebuah sebuah arsitertur jaringan baru dimana kontrol jaringan dipisahkan dari forwarding dan diprogram secara langsung [7].

Pada gambar 1, digambarkan bagaimana logical dari SDN arsitektur, dimana merupakan jaringan pintar yang secara logical tersentralisasi berdasarkan *software (software-base)*, selain itu ONF menyatakan bahwa dengan SDN tidak lagi membutuhkan protokol standar, tetapi cukup hanya menerima intruksi dari sebuah SDN kontroler.



Gambar 1. Arsitektur SDN (Sumber: ONF 2012)

Switch OpenFlow

Switch openFlow tediri dari dua jenis, yang pertama adalah *hardware-based switch*, yang telah dijual secara komersial oleh beberapa vendor. *Switch* jenis ini telah memodifikasi *hardware*-nya, menggunakan TCAM *(Ternary Content Addressable Memory)* dan menggunakan OS khusus untuk mengimplementasikan *Flow-Table* dan protokol *OpenFlow*. Jenis yang kedua adalah *software-based switch* yang menggunakan sistem UNIX / Linux untuk mengimplementasikan seluruh fungsi switch OpenFlow [8].

Terdapat beberapa perbedaan mendasar dari switch komersial (switch biasa) dengan switch openflow, dalam thesis Chung Yik,EE.,, perbedaan tersebut seperti terlihat pada tabel 1 dibawah ini; [4]

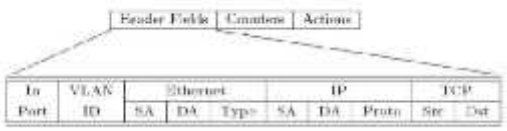
Tabel 1. Tabel Perbandingan switch menurut Chung Yik, EE

Switch Openflow	Switch komersial (switch biasa)
Terpisahnya *control path* dan *data path*	*Control path* dan *data path* terletak pada perangkat yang sama (tidak terpisah)
Memungkinkan terjadi inovasi dalam jaringan	Membatasi inovasi dalam jaringan
Menyediakan *platform* yang dapat diteliti dan diujicobakan pada jaringan sesungguhnya.	Tetap dan sulit untuk diujicobakan (dibuat tetap oleh *vendor*)
Fungsi yang dapat didefiniskan oleh user (dapat diprogram)	Arsitektur tertutup sehingga tidak dapat diprogram ulang
Setiap keputusan untuk melakukan pengiriman dilakukan oleh kontroler	Mengirimkan semua paket yang diterima keluar dari switch

Protokol *OpenFlow*

Protokol openFlow adalah sebuah standar terbuka yang memungkinkan peneliti melakukan kontrol langsung pada jalannya paket data yang akan di routing kan pada jaringan. OpenFlow, adalah konsep yang sederhana, openflow mela-kukan sentralisasi terhadap kerumitan dari jaringan kedalam sebuah *software* kontroler, sehingga seorang administrator dapat mengaturnya dengan mudah, hanya mengatur kontroler tersebut. McKeown, N. dkk, memperkenalkan konsep OpenFlow ini dengan ide awal adalah menjadikan sebuah *network* dapat di program/ dikontrol. [9]

OpenFlow adalah protokol yang bersifat terbuka, yang menjembatani komunikasi antara perangkat jaringan dengan sebuah kontroler. Seperti layaknya sebuah protokol ethernet, OpenFlow juga memiliki *field*, seperti terlihat pada gambar 2 berikut ini.



Gambar 2. OpenFlow Tabel Fields (Sumber: Mateo, M.P. 2009)

Seperti terlihat pada gambar .3, bahwa protokol OpenFlow terdiri dari 3 *fields* yaitu *Header Fields*, *Counter* dan *Action. Header fields* adalah sebuah paket header yang mendefinisikan *flow*, *fields* nya terdiri dari enkapsulasi seperti enkapsulasi segmen pada protokol VLAN ethernet standar. *Counter* adalah sebuah *fields* yang menjaga jejak jumlah paket dan *byte* untuk setiap *flow*, dan waktu jejak paket terakhir cocok dengan *flow* (untuk membantu dalam membuang atau me-nonaktif-kan *flow*). *Action* adalah *fields* yang mendefinisikan bagaimana paket data akan diproses [8].

Kontroler

Sebuah kontroler openflow bertanggung jawab untuk menambahkan atau menghilangkan isi dari flow dari openflow table yang ada didalam perangkat openflow itu sendiri. Ada 2 tipe dari kontroler : ***Statis***, sebuah kontroler statis dapat berupa perangkat yang dapat menambahkan atau menghilangkan flow dari flow table secara statis. ***Dinamis*** sebuah kontroler dinamis secara dinamis memanipulasi isi dari flow sehingga cocok untuk beberapa konfigurasi. [10]

Floodlight merupakan kontroler yang digunakan dalam pengembangan proyek *SDN (Software-Defined Network*). Floodlight berlisensi *Apache* dan berbasis JAVA. Floodlight dirancang untuk bekerja dengan meningkatnya jumlah *switch*, *router*, *switch virtual* dan jalur akses yang mendukung standar Openflow [11]. Gambar 3 dibawah ini, adalah *framework* dari kontroler floodlight.

Gambar 3. Floodlight framework (Sumber: http://www.projectfloodlight.org/floodlight/)

Ada dua tingkah laku kontroler dalam menangani *flow*, yaitu *reactive* dan *proactive*. Adapun pengertiannya adalah sebagai berikut:

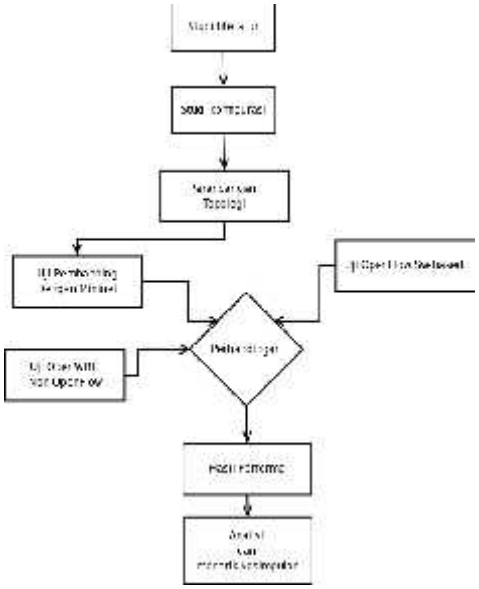
Reactive, kontroler dirancang untuk tidak melakukan apa-apa sebelum paket pertama diterima. Ketika kontreler menerima pesan pertama, memperkenalkan aturan tersebut ke tabel *flow* swicth untuk memerintahkan switch melakukan *forwarding* paket. Setiap perubahan *flow* yang kecil

sekalipun menimbulkan tambahan waktu *setup flow*. Jika koneksi kontrol hilang, switch terbatasi kemampuannya dan menunggu kontroler mengirimkan aturan yang baru untuknya [12]. Akan ada penundaan *(delay)* kecil terhadap kinerja pada setiap paket yang akan diteruskan. Hal ini sama dengan pada saat paket pertama diterima oleh switch dimana switch harus menunggu aturan yang diberikan oleh kontroler.

Proactive, kontroler melakukan pra-populasi tabel *flow* dalam *switch.* Tambahan waktu melakukan *setup* pada tabel *flow* tidak ada. Kehilangan koneksi kontrol tidak mengganggu lalu lintas data[12].

Metode Penelitian

Dalam penelitian ini, metode penelitian yang akan dilakukan oleh penulis adalah sebagai berikut: (1) Studi literatur, bahan atau materi penelitian yang dikumpulkan berupa literatur/referensi yang berhubungan dengan penelitian yang akan dilaksa-nakan. (2) Studi konfigurasi, konfigurasi *hardware* dan *software* dari kontroler yang akan digunakan, baik cara konfigurasi, spesifikasi dan troubleshoot-nya. (3)Topologi dirancang dengan switch yang tersedia. (4)Uji pembanding, Uji performa dengan mininet emulator dilakukan sebagai perbandingan hasil dari uji switch yang akan dilakukan. (5) Pengujian, dilakukan pengujian latency dengan melakukan ping dengan besar paket ICMP tertentu ke masing-masih host, sebelum dan sesudah dijalankan OpenFlow protokol. (6) Pengujian performa prototipe dengan jperf pada protokol TCP dan UDP, hasil uji dengan besar paket tertentu dan membandingkannya dengan arsitektur jaringan tanpa SDN. (7) Hasil dibandingkan dengan hasil yang didapat dari uji pembanding yang dilakukan dengan emulator mininet. Alur penelitian ini dapat dilihat pada gambar 4 dibawah ini.



Gambar 4. Alur penelitian

Gambaran Umum Penelitian

Penelitian ini secara umum terdiri dari beberapa langkah, yaitu simulasi menggunakan emulator mininet sebagai uji pembanding (dianggap sebagai representatif dari *dedicated switch openflow*), pengujian performa switch OpenWRT yang belum di beri agen OpenFlow (switch non OF), pengujian performa *switch OF software-based.*

Emulator mininet

Mininet adalah perangkat lunak emulator jaringan yang dapat digunakan untuk mensimulasikan jaringan, baik switch, *host*, dan kontroler SDN hanya dalam satu sumberdaya PC/laptop dalam satu perintah. Mininet dapat melakukan simulasi jumlah *node* (dalam hal ini switch) yang banyak sesuai dengan kebutuhan dari penelitian.

Pada penelitian ini, mininet diinstal dan dikonfigurasikan pada VirtualBox (mesin virtual) sehingga kontroler akan berada pada PC yang berbeda dengan mininet. Pada penelitian ini, kontroler diletakkan/diinstal pada PC/laptop yang juga menjalankan VirtualBox. Mininet digunakan sebagai representatif dari *dedicated switch openflow (hardware-based).*

Hasil dari simulasi mininet dengan variabel yang ada dan dibandingkan dengan hasil dari *switch OF software-based* yang dibuat, baik dari hasil pengujian *latency* dan *througput* dengan switch yang belum diberikan agen openflow dan diperbandingkan pula dengan hasil uji pembanding yang dilakukan dengan emulator mininet.

Rancangan topologi

Topologi yang digunakan pada penelitian ini adalah topologi linear yang melibatkan 2 buah *switch OF software-based* yang terhubung dengan sebuah kontroler. Topologi yang digunakan dapat dilihat pada gambar 5 dibawah ini.

Gambar 1. Rancangan topologi

Topologi pada gambar 5 adalah topologi yang digunakan pada simulasi mininet dan topologi ini pula yang akan digunakan sebagai topologi pada prototipe.

Keterangan gambar :

C0	adalah kontroler ke 0 yang merupakan kontroler tersentralisasi yang melakukan kontrol terhadap kedua buah switch.
hosts	adalah host yang terkoneksi langsung ke switch
Sw_OpenFlow	adalah switch OpenFlow, kedua buah switch tersebut, terhubung satu sama lain, dan kedua switch terhubung langsung dengan kontroler yang dihubungkan dengan sebuah hub.
———	Koneksi antar perangkat pada jaringan OpenFlow
.....	Koneksi pada jaringan kontroler (koneksi dari kontroler ke switch)

Uji pembanding dengan *emulator mininet*

Topologi yang telah dirancang sebelumnya diujikan pada emulator mininet. Pengujian ini dimaksudkan untuk mendapatkan data pembanding nilai latency dan throughput dengan data pada switch OF software-based yang akan diujikan.

Topologi yang akan diujikan terlihat pada scrip dibawah ini dan disimpan dengan nama file ujibanding.py,

```
from mininet.topo import Topo

class MyTopo( Topo ):

   "Topologi prototipe."
  def __init__( self ):

     "Membuat topologi untuk uji pembanding."
     # Initialize topology
     Topo.__init__( self )

     # Add hosts and switches
     left0Host = self.addHost( 'h1', ip='192.168.10.1/24' )
     right0Host = self.addHost( 'h2', ip='192.168.10.2/24' )
     left1Host = self.addHost( 'h3', ip='192.168.10.3/24' )
     right1Host = self.addHost( 'h4', ip='192.168.10.4/24' )
     leftSwitch = self.addSwitch( 's1' )
     rightSwitch = self.addSwitch( 's2' )

     # Add links
     self.addLink( left0Host, leftSwitch )
     self.addLink( left1Host, leftSwitch )
     self.addLink( leftSwitch, rightSwitch )
     self.addLink( rightSwitch, right0Host )
     self.addLink( rightSwitch, right1Host )

topos = { 'topoujibanding': ( lambda: MyTopo() ) }
```

Scrip tersebut dijalankan pada emulator mininet dengan kontroler diletakkan diluar virtualbox yang dijalankan oleh mininet. *Scrip* dijalankan dengan perintah dibawah ini.

```
mininet@mininet#sudo mn --custom ujibanding.py --topo topoujibanding --controller=remote, ip=192.168.1.76, port=6633 --mac --link tc,bw=100
```

Mininet dijalankan pada Laptop dengan spesifikasi sebagai berikut :
- CPU : Intel® Pentium(R) CPU 987 @ 1.50GHz × 2
- RAM : SODIMM DDR3 2Gb 1333MHz
- Hardisk : 500 Gbyte
- NIC : Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit ethernet Controller (rev 0a)
- Operation System : Linux UBUNTU 12.04 LTS kernel 3.13.0-43-generic.

Kontroler floodlight telah dijalankan terlebih dahulu, dengan perintah,

```
$ java -jar floodlight.jar
```

GUI dari floodlight ini dapat diakses pada web browser dengan alamat, seperti pada gambar 6 dibawah ini.
http://localhost:8080/ui/index.html

Gambar 2. GUI Floodlight

Pengujian *latency* dan *throughput* pada *mininet* sebagai pembanding

Pengujian latency yang dilakukan pada mininet dilakukan pada besar paket ICMP dari 64 byte hingga 8192 byte. Pengujian dilakukan dengan pengu-langan sebanyak 15 kali pada setiap paket data ICMP yang dikirimkan.Pengujian throughput yang dilakukan pada mininet dilakukan pada protokol TCP dengan TCP windows size 128 kBps hingga 1024kBps dan UDP Buffer size dan Bandwidth sebesar 128 kBps hingga 1024kBps. Pengujian menggunakan jperf sebagai tool penguji. Pada sisi server, jperf diseting dengan konfigurasi

dimana besar windows sizes diberikan variabel mulai dari 1024kBps.

iperf -s -P 0 -i 1 -p 5001 -w 1024K -f k

Untuk UDP digunakan seting mulai dari packet size yang sama juga.

iperf -s -u -P 0 -i 1 -p 5001 -w 1024K -f k

Sedang pada sisi client, jperf diseting dengan konfigurasi

iperf -c 10.0.0.1 -P 1 -i 1 -p 5001 -w 1024K -f K -t 10

Untuk UDP pada sisi client digunakan seting sebagai berikut

iperf -c 10.0.0.1 -u -P 1 -i 1 -p 5001 -w 1024K -f k -b 1024M -t 10 -T 1

Pengujian *latency* dan *troughput* pada pada *switch* sebelum diberikan agen *OpenFlow*

Topologi yang telah dirancang sebelumnya diujikan pada switch yang belum diberikan agen Openflow. Pengujian ini dimaksudkan untuk mendapatkan data latency dan throughput dengan data pada switch yang belum diberi agen Openflow. Pengujian latency yang dilakukan pada switch yang belum diberi agen Openflow dilakukan pada besar paket ICMP dari 64 byte hingga 8192 byte. Pengujian dilakukan dengan pengulangan sebanyak 15 kali pada setiap paket data ICMP yang dikirimkan. Pengujian throughput yang dilakukan pada switch yang belum diberi agen Openflow dilakukan pada protokol TCP dengan TCP windows size sebesar 1024kBps dan UDP Buffer size dan Bandwidth sebesar 1024kBps.

Pengujian *latency* dan *troughput* pada pada *switch switch OF Software-based*

Spesifikasi hardware dari switch TP-Link WR1043ND versi 1.11 seperti terlihat pada gambar 7 yang diubah firmware-nya dengan firmware yang mempunyai agen Openflow dapat dilihat pada tabel 2 berikut. Port yang ada pada switch setelah mengalami perubahan firmware dapat dilihat pada gambar 8 dan tabel 3 berikut.

Gambar 3. TP-Link WR1043ND Versi 1.11

Tabel 2. Spesifikasi hardware switch

Type	HW ver. 1.x
Instruction set:	MIPS32
Vendor	Qualcomm Atheros
Hardware	[illegible]
System-On-Chip:	AR9132 rev 2 (MIPS 34Kc V7.4)
CPU freq	34Kc V7.4 400 MHz
Target name	ar71xx
Flash-Chip.	ST 25P64V6P
Flash size:	8192 KiB
RAM:	32 MiB
Wireless:	Atheros AR9103 2.4 GHz 802.11bgn
Wireless Power:	Maximal power output is 24 dBm (251 mW)
802.11n:	3 x 3 : 2
Switch:	[illegible]
USB:	Yes 1 x 2.0 (OHCI platform, device name 2-1)
Power:	DV 12 V 1.5 A
Serial:	Yes
JTAG:	Yes

Gambar 4. Port pada switch yang telah dirubah firmware-nya

Tabel 3. Konfigurasi port

Switch	IP address	Port Ke kontroler	Port terkoneksi
Switch (PID01)	192.168.1.11/24	Eth0.5	Eth0.1, eth0.3, eth0.4
Switch (PID02)	192.168.1.12/24	Eth0.5	Eth0.1, eth0.3, eth0.4

Pengujian pada *switch OF software-base* ini, dilakukan dengan varibel yang sama pada pengujina sebelumnya.

Hasil dan Pembahasan

Dari data yang diperoleh, baik data dari uji pembanding, uji terhadap switch non OF dan switch OF software-based. Data yang diperoleh dilakukan analisis pada nilai latency dan throughput yang dihasilkan.

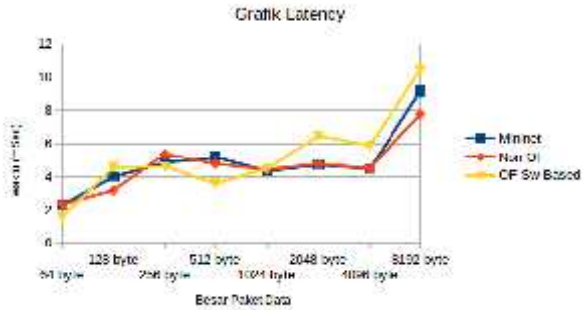
Pengujian Latency

Pada tabel 4 dan gambar 9 dibawah ini, data nilai rata-rata latency yang dihasilkan pada setiap pengujian. Nilai rata-rata latency pada setiap pengujian.

Tabel 4. Nilai rata-rata latency pada setiap pengujian

	64 byte	128 byte	256 byte	512 byte	1024 byte	2048 byte	4096 byte	8192 byte
Mininet	2.288	3.999	4.855	5.191	4.341	4.726	4.477	9.149
Non OF	2.341	3.193	5.376	4.787	4.436	4.772	4.515	7.776
OFSw-Based	1.622	4.545	4.613	3.585	4.499	6.48	5.89	10.421

Diperoleh hasil bahwa switch OF software-based mengalami latency yang sebanding (hampir sama) dibandingkan dengan mininet maupun switch yang belum diberi agen openflow (switch non OF). Walaupun pada besar paket data ICMP tertentu *switch openflow software-based* mengalami keterlambatan yang lebih tinggi dibanding dengan dua pembanding lainnya, seperti dapat dilihat pada gambar 9 dibawah ini.



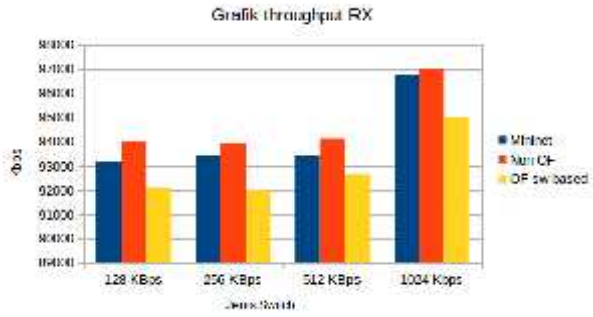
Gambar 5. Grafik latency

Pengujian TCP *Throughput*

Pengujian *throughput* dengan protokol TCP yang diperoleh hasil *bandwidth* dari mininet, switch non OF dan *switch OF software-based* seperti pada tabel 5 dan gambar 10 dibawah ini. Data yang digunakan adalah data Rx (Receive).

Tabel 5. Nilai TCP throughput RX

	128 Kbps	256 Kbps	512 KBps	1024 Kbps
Mininet	93216	93400	93392	**96771**
Non OF	94012	93898	94112	97011
OF sw-based	92131	92001	92661	95012
Gap rata-rata				
OF sw-based vs mininet			**OF sw-based vs non OF**	
-1243.5			-1807	



Gambar 6. Grafik TCP Throughput Rx

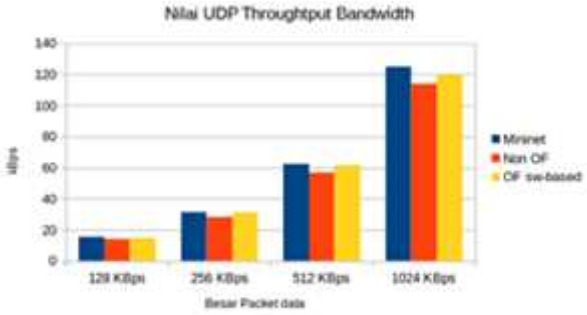
Dari grafik yang terlihat pada gambar 10 diatas, terlihat bahwa throughput protokol TCP yang diperoleh switch OF software-based diperoleh hasil mendekati nilai dari kedua switch pembanding lainnya, walau nilai yang dihasilkan adalah nilai yang paling kecil dibandingkan dengan switch non OF bahkan terhadap mininet. Nilai tertinggi diperoleh switch non OF, dikarnakan penerusan data (data forwarding) yang dilakukan switch menggunakan control plane yang melekat pada switch dan tidak membutuhkan sumberdaya yang berbeda. Hal ini tidak dapat diartikan bahwa switch OF software-based tidak memiliki throughput yang baik pada protokol TCP. Nilai gap throughput rata-rata switch OF software-based lebih rendah 1807kbps dibanding switch non OF dan lebih rendah 1243.5kbps dibanding mininet, nilai throughput dapat saja menunjukkan nilai yang berbeda pada jenis switch yang berbeda.

Pengujian UDP throughput

Nilai uji throughput pada protokol UDP diperoleh 2 nilai yaitu besar bandwidth dan nilai jitter. Tabel 6 merupakan tabel yang menunjukkan perbedaan bandwidth yang dihasilkan switch pada protokol UDP. Pada gambar 11, dapat dilihat bahwa *switch OF software-based* nilai throughput protokol UDP menunjukkan nilai yang lebih tinggi dibandingkan switch non OF. Walaupun nilai ini masih dibawah nilai uji pembanding (mininet).

Tabel 6. Nilai UDP throughtput Bandwitdth

	128 KBps	256 KBps	512 KBps	1024 KBps
Mininet	15.6	31.3	62.5	125
Non OF	14.3	28.3	56.9	114.01
OF sw-based	15.1	31	61.6	120
Gap rata-rata				
OF sw-based vs mininet			**OF sw-based vs non OF**	
-1.68			3.55	



Gambar 7. Grafik Nilai UDP Throughput bandwidth

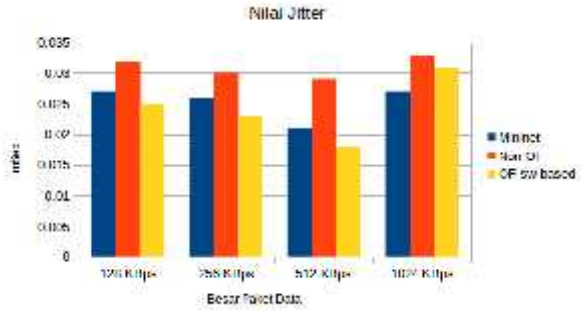
Nilai gap *throughput bandwidth* protokol UDP rata-rata *switch OF software-based* lebih tinggi 3.55kBps dibanding switch non OF dan lebih rendah 1.68kBps dibanding mininet. Nilai yang dihasilkan, menunjukkan *switch OF software-based* relatif lebih baik penyampaian protokol UDP dibandingkan dengan switch non OF walau sedikit lebih rendah dibandingkan dengan mininet.

Untuk nilai jitter pada ketiga switch dapat dilihat pada tabel 7 dan gambar III-8 dibawah ini.

Tabel 7. Nilai Jitter

	128 KBps	256 KBps	512 KBps	1024 KBps
Mininet	0.027	0.026	0.021	0.027
Non OF	0.032	0.03	0.029	0.033
OF sw-based	0.025	0.023	0.018	0.031

Gap rata-rata	
OF sw-based vs mininet	**OF sw-based vs non OF**
-0.001	-0.008



Gambar 8. Nilai Jitter

Nilai jitter yang diperoleh menunjukkan bahwa *switch OF software-based* memberikan nilai yang baik dibandingkan pembandingnya pada besar paket data 129 kBps hingga 512kBps, namun mengalami penundaan lebih tinggi pada besar paket 1024kBps. Nilai gap *throughput* jitter protokol UDP rata-rata *switch OF software-based* lebih rendah 0.008ms dibanding switch non OF dan lebih rendah 0.001ms dibanding mininet. Nilai jitter pada pengujian throughput protokol UDP dapat dikatakan menghasilkan nilai yang sama dengan pembandingnya.

Kesimpulan dan Saran

Dari data pengujian yang dilakukan dapat ditarik kesimpulan bahwa :

Performa *switch OF software-based* dapat dikatakan baik dengan hasil gap rata-rata pada setiap pengujian menunjukkan angka yang tidak tinggi. Bahkan pada pengujian jitter dapat dikatakan sama.

1. Pada pengujian *latency*, *switch OF software-based* mempunyai hasil yang sebanding/setara dengan pembandingnya (switch non OF) dan dengan uji pembanding (mininet) walaupun pada pengujian tertentu mengalami peningkatan.
2. Pada pengujian *throughput* protokol TCP, *switch OF software-based* memberikan hasil yang rendah dibandingkan dengan pembandingnya, dengan Nilai gap rata-rata switch OF software-based lebih rendah 1807kbps dibanding switch non OF dan lebih rendah 1243.5kbps dibandingkan dengan mininet. Performa *switch OF software-based* rendah pada *throughput* protokol TCP.
3. Pada pengujian *throughput* protokol UDP, bandwidth yang diperoleh nilai gap *throughput bandwidth* protokol UDP rata-rata *switch OF software-based* lebih tinggi 3.55kBps dibanding switch non OF dan lebih rendah 1.68kBps dibanding mininet. Performa *switch OF software-based* lebih baik pada *throughput* protokol UDP.
4. Pada pengujian jitter protokol UDP, nilai gap jitter protokol UDP rata-rata s*witch OF software-based* lebih rendah 0.008ms dibanding switch non OF dan lebih rendah 0.001ms dibanding mininet. Nilai jitter pada pengujian *throughput* protokol UDP dapat dikatakan menghasilkan nilai yang sama dengan pembandingnya.
5. Dengan melihat hasil yang diperoleh, *switch OF software-based* dapat digunakan untuk menggantikan *dedicated openflow switch* untuk mengimplementasikan SDN baik pada sekala menengah dan kampus.

Untuk melanjuti dan memperbaiki penelitian ini, penulis memberikan saran sebagai berikut:

1. Pengembangan *switch OF software-based* menggunakan switch berbasis *firmware* yang berbeda atau dapat pula dengan jenis yang berbeda.
2. Penambahan variabel uji yang lebih detail dan banyak.
3. Dapat dilakukan pengujian dengan menggunakan kontroler yang berbeda.

Daftar Pustaka

[1] Applemen,M.; De Boer,M., 30 Mei 2013, Performance Analysis of *OpenFlow* Hardware, University Of Amsterdam 2012, http://staff.science.uva.nl/~delaat/rp/2011-2012/p18/report.pdf

[2] Shirazipour,M.; John, W.; Kempf, J.; Green, H.; Tatipamula, M., 2012, Realizing Packet-Optical Integration with SDN and OpenFlow 1.1 Extensions, Communications (ICC), 2012 IEEE International Conference on ISSN:1550-3607

[3] Sherwood, R., 30 Mei 2013, An Experimenter's Guide to *OpenFlow*, GENI Engineering Workshop June 2010, http://www.deutsche-telekom-laboratories.de/~robert/GENI-Experimenters-Workshop.ppt

[4] Chung Yik,EE., 2012, IMPLEMENTATION OF AN OPEN FLOW SWITCH ON NETFPGA, Faculty of Electrical Engineering , Universiti Teknologi Malaysia

[5] Kartadie, R., 2014, PROTOTIPE INFRASTRUKTUR SOFTWARE-DEFINED NETWORK DENGAN PROTOKOL OPENFLOW MENGGUNAKAN UBUNTU SEBAGAI KONTROLER, Program Magister Teknik Informatika Program Pascasarjana STMIK AMIKOM Yogyakarta.

[6] Casado,M.; Michael J.; Freedman ; Pettit, J.; Luo, J.; McKeown, N.; Shenker,S., 2007, Ethane: Taking Control of the Enterprise, SIGCOMM'07,

Kyoto,Japan. Copyright 2007 ACM 978-1-59593-713-1 /07/0008

[7] Anonim, 10 Mei 2013, Software-Defined Networking: The New Norm for Networks, Open Networking Foundation. 2012, https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[8] Mateo, M.P., 2009, OpenFlow Switching Performance, III Facoltà di Ingegneria dell'Informazione Corso di Laurea in TelecommuNICation Engineering, POLITECNICO DI TORINO, Italy

[9] McKeown,N.; Anderson,T.; Balakrishnan,H.; Parulkar,G.; Peterson, L.; Rexford,J.; Shenker,S.; Turner,J.; 1 Mei 2013, OpenFlow: Enabling Innovation in Campus Networks, Stanford University 2008, www.OpenFlow.org/documents/OpenFlow-wp-latest.pdf

[10] Cohen, M., 10 Desember 2014, Software-Defined Networking and the Floodlight controller, 2012, http://www.internet2.edu/presentations/jt2012summer/20120717-Cohen-Floodlight.pdf

[11] Muntaner , G. R.T., 9 April 2013, Evaluation of OpenFlow Controllers , 15 Oktober 2012, http://www.valleytalk.org/wp-content/uploads/2013/02/Evaluation_Of_OF_Controllers.pdf

[12] Mendonca, M.; Nunes, B.A.A.; Nguyen, X.; Obraczka, K.; Turletti, T., 11 April 2014, A Survey of Software-Defined Networking: Past, Present, and Future of Pragrammable Networks, hal-00825087,version2, http://hal.archives-ouvertes.fr/docs/00/83/50/14/PDF/bare_jrnl.pdf